

CS202 HOMEWORK#3

1. Questions

a) Insertion in an AVL Tree

insert 13 -> 13

insert 6 -> 13
/

6

insert 3 -> 6
/ \

3 13

insert 7 -> 6
/ \

3 13
/

7

insert 2 -> 6
/ \

3 13
/ \

2 7

insert 4 -> 6
/ \

3 13
/ \ /

2 4 7

insert 11 -> 6
/ \

3 11
/ \ / \

2 4 7 13

insert 0 -> 6
/ \

3 11
/ \ / \

2 4 7 13
/

0

insert -1 -> 6
/ \

2 11
/ \ / \

0 3 7 13
/ \

-1 4

insert 1 -> 6
/ \


2 11
/ \ / \

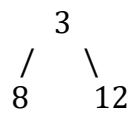
0 3 7 13
/ \ \

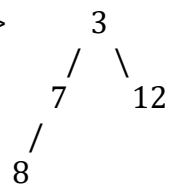
-1 1 4

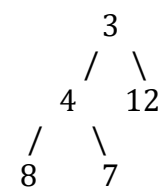
b) Insertion & deletion in a heap

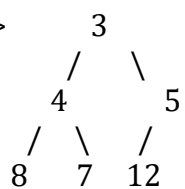
insert 12 -> 12

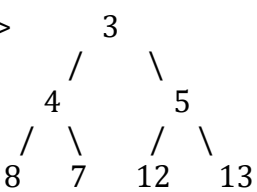
insert 8 -> 

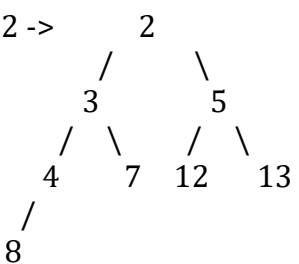
insert 3 -> 

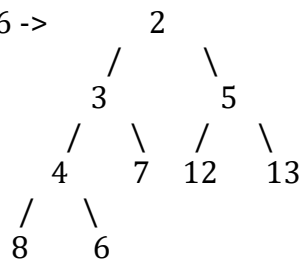
insert 7 -> 

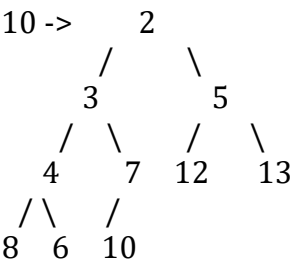
insert 4 -> 

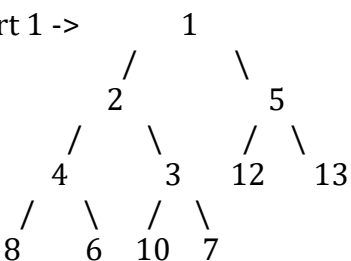
insert 5 -> 

insert 13 -> 

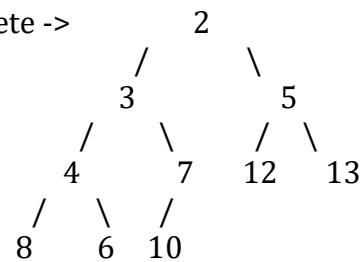
insert 2 -> 

insert 6 -> 

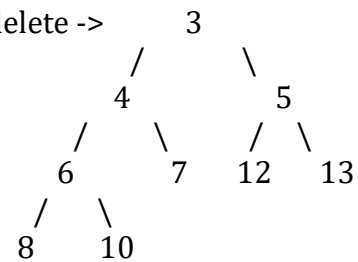
insert 10 -> 

insert 1 -> 

delete ->

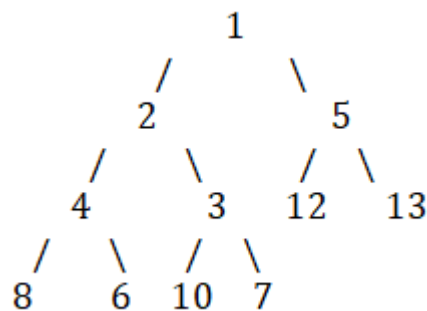


delete ->



c) Traversals in a heap

Consider the binary heap in part b after insert 1.



Traversals for this heap are the following:

Preorder: 1 2 4 8 6 3 10 7 5 12 13

Inorder: 8 4 6 2 10 3 7 1 12 5 13

Postorder: 8 6 4 10 7 3 2 12 13 5 1

The output for preorder traversal of this heap is not sorted. However, first $(h_{\text{left}} + 1)$ items in the output are sorted where h_{left} is the height of the left subtree. In general, first $\lfloor \log_2 n \rfloor + 1$ items of the output are sorted, where n is the size of the heap. If the heap is a minheap, they are sorted in ascending order. If the heap is maxheap, they are sorted in descending order. For other traversals, the output is not sorted.

The reason for this is that for preorder traversal, first $k = \lfloor \log_2 n \rfloor + 1$ items will be always a node with two children or a leaf node. Since a heap is defined so that a parent node is larger/smaller than their children, for the first k items, we always get one of the following, depending on the type of the heap:

$$a_0 \geq a_1 \geq a_2 \geq \dots \geq a_k (\text{max heap})$$

$$a_0 \leq a_1 \leq a_2 \leq \dots \leq a_k (\text{min heap})$$

Postorder traversal is not sorted because there is no relation between the left child and the right child of a heap. For inorder traversal, the relation between items will be like the following but it won't be sorted at all:

$$a_i \leq a_{i+1}, a_{i+1} \geq a_{i+2} (\text{max heap})$$

$$a_i \geq a_{i+1}, a_{i+1} \leq a_{i+2} (\text{min heap})$$

, where i is an even numbered index of the inorder traversal. In other words, an item with an odd index will be larger than both its predecessor or successor if the heap is a minheap, and vice versa. Though, it doesn't mean that the array is sorted.

d) Minimum number of nodes in an AVL tree

The expression for minimum number of nodes in an AVL tree is shown as $N(h) = N(h-1) + N(h-2) + 1$; where h is the height of the tree. $N(h-1)$ is the # nodes for the higher subtree in the AVL tree and $N(h-2)$ is the # nodes for the shorter subtree in the AVL tree. Since we want the # nodes to be minimum, we define $N(h)$ recursively so that AVL tree property is still conserved.

$$N(h) = N(h-1) + N(h-2) + 1$$

$$N(h)^{(p)} = -1$$

$$N(h)^{(h)} = \alpha \left(\frac{1+\sqrt{5}}{2} \right)^h + \beta \left(\frac{1-\sqrt{5}}{2} \right)^h$$

$$\alpha = \frac{3\sqrt{5}+5}{10}, \beta = \frac{-3\sqrt{5}+5}{10}$$

$$N(h) = \left(\frac{3\sqrt{5}+5}{10} \right) \left(\frac{1+\sqrt{5}}{2} \right)^h - \left(\frac{3\sqrt{5}-5}{10} \right) \left(\frac{1-\sqrt{5}}{2} \right)^h - 1$$

Since $N(h)$ is a nonhomogeneous recurrence relation, we were able to find the exact equation for it. Now, we can find $N(15)$ to find minimum number of nodes if the height is 15.

$N(15) = 1596,9998 - 0,0001 - 1 \approx 1596$ is the minimum number of nodes of an AVL tree of height 15.

e) Pseudocode

```
If (tree is empty)

    If (root index is out of bounds)

        Bst is not complete, therefore not a min-heap

    Else

        Bst is a min-heap

Else if (node is smaller than its children)

    Bst is a min-heap if the left and right subtree
    is also min-heap AND the left and right subtree
    is also complete BST

Else

    Bst is not a min-heap
```

The boolean “node is smaller than its children” will return false only when the node has at least one child and the node is greater than at least one of them.

2. Report

The heap class in this assignment uses the array implementation. It has two properties, a list of items (`int items[MAX_SIZE]`) and size of the list. `MAX_SIZE` is the largest the heap can get. No resizing was implemented in this class if there was an insertion when the heap was full. It has one more constructor and another `popMaximum` function.

The constructor takes an array and builds a heap out of that array. It also counts the number of key comparisons during the constructions. Calculations show that the number of key comparisons for heapsort algorithm is bounded by $n \log n$. Below is the actual number of key comparisons for heapsort algorithm:

Zübeyir Bodur
21702382

Size = 1000	# Key Comparisons: 26128
Size = 2000	# Key Comparisons: 58447
Size = 3000	# Key Comparisons: 92576
Size = 4000	# Key Comparisons: 128536
Size = 5000	# Key Comparisons: 165370

If we divide the each number of key comparisons to their size, we should get numbers that scale as fast as $\log_2 n$.

$$26128/1000 = 26.128$$

$$58447/2000 = 29.2235$$

$$92576/3000 = 30.8586$$

$$128536/4000 = 32.134$$

$$165370/5000 = 33.074$$

If the number of key comparisons was X , and if we know X/n is $O(\log n)$, we can conclude that X is $O(n \log n)$.